CONF-830576--7

Submitted to the Third Biennial Conference on Real-Time Computer Applications in Nuclear & Particle Physics, Berkeley, CA
May 16-19, 1983

BNL 33165

BNL--33165

DE83 014434

BROOKHAVEN FASTBUS/UNIBUS INTERFACE

G. Benenson, J. Bauernfeind, R.C. Larsen, L.B. Leipuner, and W.M. Morse*
Brookhaven National Laboratory
Associated Universities, Inc.
Upton, New York 11973

R.K. Adair, J.K. Black, S.R. Campbell, H. Kasha, and M.P. Schmidt
Yale University New Haven, Connecticut 06511

I. Summary

A typical high energy physics experiment requires both a high speed data acquisiton and processing system, for data collection and reduction; and a general purpose computer to handle further reduction, bookkeeping and mass storage. Broad differences in architecture, format or technology, will often exist between these two systems, and interface design can become e formidable task.

The PDP-11 series minicomputer is widely used in physics research, and the Brookhaven FASTBUS is the only standard high speed data acquisition system which is fully implemented in a current high energy physics experiment. This paper will describe the design and operation of an interface between these two systems. The major issues are elucidated by a preliminary discussion on the basic principles of Bus Systems, and their application to Brookhaven FASTBUS and UNIBUS.

II. Whys and Wherefores of Busses

The simplest data transmission system would include a MASTER device, a SLAVE device and an interconnecting link. (The terms "Master" and "Slave" have come into general use in spite of their negative human connotations. They are, however, technically preferable to "Driver"/"Receiver" or "Talker"/"Listener", etc. because a "Master" may initiate a "Read" Operation in which the "Slave" is actually the sender of the data.) At a minimum, two kinds of information must appear on the connecting link: the DATA itself, and DATA TRANSFER CONTROL information which is necessary both to specify the direction of data flow, and to synchronize its transmission and reception. (See Fig. 1A.)

In a larger system, use of individual links between all potential Master-Slave pairs could in principle support very high speed parallel operation. However, the resulting network would require vast amounts of cable and connectors, and impose a major arbitration problem on each module. In practical terms, the cost of high-speed micro-electronics has been declining both absolutely and relative to the cost of connections. In addition, master and slave devices may often be collected into groups in which data transfers must occur in sequence: in other words, where there would be no benefit from concurrent transmissions in a network. (See Fig. 1b.)

For all of the reasons mentioned, it is desirable to use interconnecting lines which are shared among many potential masters and slaves. These lines are commonly multiplexed as a function of time between successive master-slave pairs, end are referred to collectively as a BUS. (See Fig. 1c.) Note that a single physical device may appear as both "Master" and "Slave" at different times. Bus interconnections require at least two additional types of information besides DATA and DATA TRANSFER CONTROL. ARBITRATION CONTROL signals are needed to determine which of many potential masters will actually assume mastership during a particular time slot on the bus. ADDRESS information is used to select which of many potential slaves will actually respond to the current master.

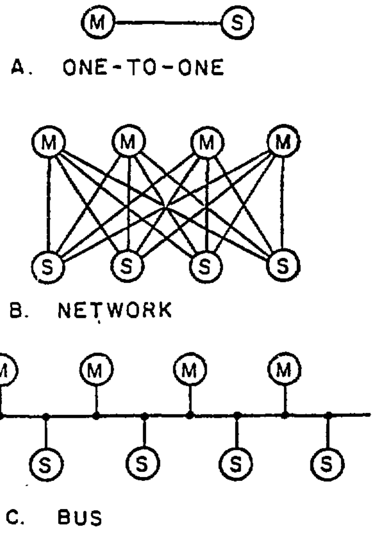

Fig. 1.

Data Transport Structures

A fifth type of information, though not logically essential, is very convenient to include in practice. SYSTEM CONTROL lines are used to force the system into a well-known initial condition, and to broadcast other types of system-wide information, such as power failure, real-time clock, etc.

The interface described below serves as a link between two very general bus structures: Brookhaven FASTBUS and UNIBUS. When a master on a UNIBUS requires a FASTBUS operation, it addresses the interface as a slave on UNIBUS. The interface then becomes a master on FASTBUS, and completes the transaction after addressing the desired FASTBUS slave. The reverse transaction is initiated by a master on FASTBUS; in this case, the interface appears as a slave to FASTBUS and a master to UNIBUS. (See Fig. 2.) It is obvious from the diagram that if FASTBUS modules and UNIBUS modules could communicate directly, there would be no need for an interface!

Why is there a multiplicity of bus structures? Specifically, why are there both a UNIBUS and a Brookhaven FASTBUS? The answers are both historical and technical. UNIBUS was introduced by DEC in 1970 to support its PDP-11 line of minicomputers, and it reflects the state of technology and architectural thinking of that time. Its continued use is partly a result of the wide profusion of UNIBUS devices. In

* Work performed under the auspices of the U.S. Department of Energy.

EJB

addition, it has been cheaper to compensate for shortcomings of the bus by adding logic circuitry than by replacing the bus itself. An example is the use of memory mapping in later PDP-11s to compensate for the lack of address space on the bus.

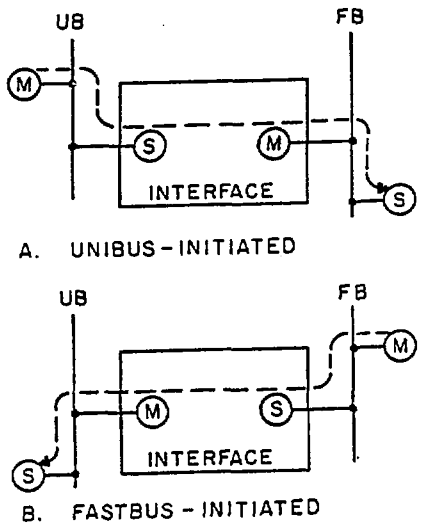

Fig. 2

Interface Function

Brookhaven FASTBUS, by contrast, is a product of the technologies and data handling needs of the Physics community of the late 1970s. It features far higher speeds, greater architectural flexibility and wider address and data spaces than does UNIBUS.

III. An Insider's View of the Busses, with a Glimpse of the Interfacing Problem

A. Data and Address Fields

Figure 3 compares the data and address fields of UNIBUS with those of FASTBUS. Figure 3A indicates that data and address are multiplexed in space on UNIBUS, i.e., they occur at the same time but in different physical space; and multiplexed in time on FASTBUS, i.e., they occur in the same space but at different times. Note that time multiplexing on FASTBUS allows the use of much larger address and data spaces within the same physical space.

Figure 3B shows how the data space of the two busses is actually used to transport data of various types. Figure 3C shows the mapping which assigns a unique address location to a segment of the data space on each bus.

Figure 3 makes evident those tasks the interface must perform in transforming the data and address fields:

1. Map the 18 bit address space of UNIBUS into the 32-bit field on FASTBUS and vice versa.
2. Convert space-multiplexed address and data fields into a time-multiplexed field, and vice versa.
3. Transform UNIBUS data types into FASTBUS types and vice versa; for example, convert 32-bit words on FASTBUS into 2 16-bit words on UNIBUS; move 8-bit FASTBUS words into the proper high- or low-order byte locations on UNIBUS, etc.
4. Generate address sequences which observe the correct mapping to data space on either bus; for example, a stream of 32-bit words occupying N FASTBUS locations will occupy 4N locations when mapped into UNIBUS address space.

B. Data Transfer Control

A convenient scheme for classifying synchronization methods is given by Levy.[1] The control signals may be periodic or aperiodic; and they may originate from the master, from both master and slave, or from some centralized location elsewhere. According to this classification, both Brookhaven FASTBUS and UNIBUS fall into the same category: their data transfers are synchronized by aperiodic signals which are issued by both master and slave. (A partial exception is the non-handshake block transfer mode on Brookhaven FASTBUS; see below.) This type of data transfer is known as a handshake: A master having data to send or receive sets a synchronization bit; the recognized slave raises a bit of its own to indicate receipt or transmission of the data; the master then lowers its bit; followed by the slave, which follows suit, completing the data transfer cycle. The two busses are also similar in their method for specifying the direction of the data transfer. In both cases there is a control bit which accompanies the address and indicates whether data is to be transferred to or from the slave (i.e., read or written).

Beyond these basic characteristics, Brookhaven FASTBUS provides several important features in its data transfer protocol which are absent from UNIBUS. Time multiplexing of address and data requires a separate handshake for each form of information. The double handshake on FASTBUS is far more powerful than the single one on UNIBUS in providing for error detection and recovery.

A variety of factors may prevent the successful completion of a data transfer within a reasonable time span. There may be no slave which recognizes the address put up by the master; or, the addressed slave may be busy, empty of data, or slow in producing or accepting it. UNIBUS provides no way for distinguishing among these possiblities; the only possible indication of an anomaly in data transfer is the failure of any slave to return the second part of the handshake.

On Brookhaven FASTBUS, by contrast, the various conditions are separable. Two options are immediately apparent: every slave may fail to return the address part of the handshake, indicating non-recognition of address; or, a slave which has completed the address handshake may fail to return the data handshake, signaling its inability to read or write data. Three other possibilties are introduced by the addition of two more bits, which may be returned by a slave with its data handshake. One of these bits shows a BUSY condition; the other, an EMPTY condition; both together are used to mark the last word of a block transfer.

Time-multiplexing of address and data lines has one inherent limitation: two fundamentally distinct transactions must occur in order to transfer a single word of data. Most time-multiplexed busses,

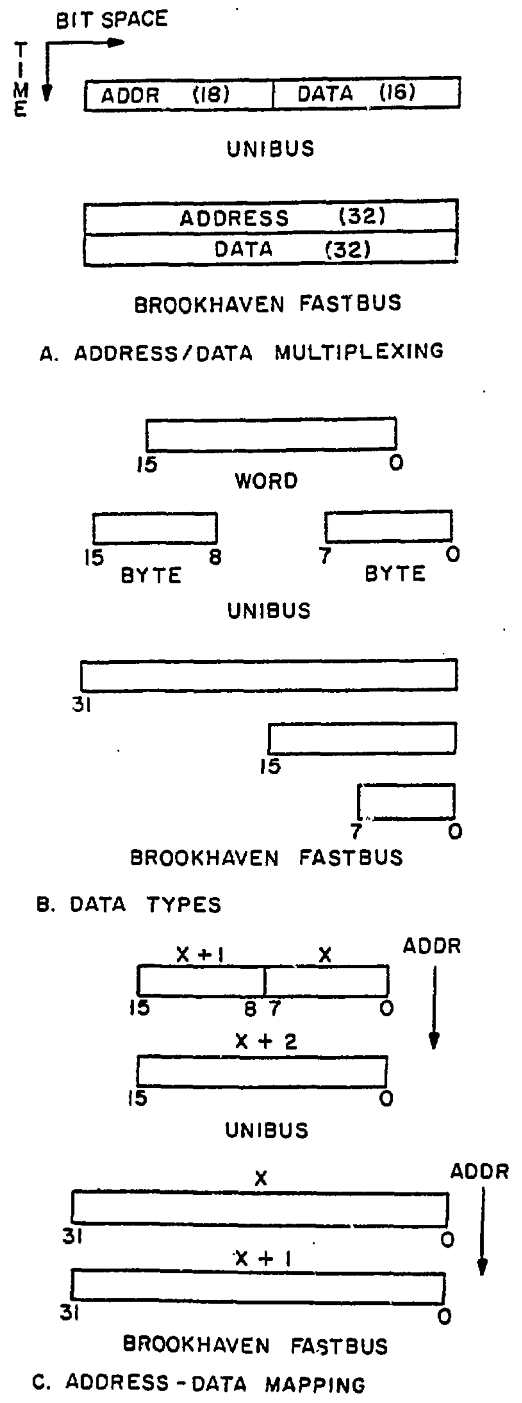

Fig. 3
Bus Comparisons

including Brookhaven FASTBUS, overcome this limitation by allowing for block transfer, in which a single address cycle is followed by multiple data cycles. The address cycle transfers the starting address; the remaining addresses are generated sequentially in the slave by counting up or down. Each data cycle of a block transfer is typically faster than the combined address/data cycle of a non-multiplexed bus, because the counting operation is inherently faster than full address recognition. Brookhaven FASTBUS implements two block transfer modes: with and without handshake. The former is, as described above, similar to a single handshaked address-word transfer followed by many handshaked data-word transfers. In "non-handshaked" mode, only the address cycle observes full handshaking. During the data cycles, only the synchronization signal issued by the master is meaningful; the return signal sent by the slave is ignored on a word-by-word basis and loss of words can be detected only on an aggregate basis. Non-handshaked block transfers sacrifice reliability to speed; because the speed of even single-word transfers on FASTBUS far outstrips that of UNIBUS, there is no reason to include non-handshake mode in the interface.

The preceding discussion of data transfer control methods on Brookhaven FASTBUS and UNIBUS suggests a number of problems the interface must solve in order to tie the two busses together:

1. Decompose a single address/data transmission, including handshake, on UNIBUS, into address transmission followed by data transmission, with double handshake, on FASTBUS, and vice versa.

2. Compose a 32-bit word on FASTBUS from 2 16-bit words on UNIBUS, and vice versa, with minimum usage of either bus.

3. Decompose a block transfer of 16- or 32-bit words on FASTBUS into a stream of 16-bit words on UNIBUS, with proper sequencing of addresses.

4. Make available to UNIBUS information from FASTBUS about the failure to complete a data transfer in the normal way.

C. Arbitration and System Control

The job of arbitration is the problem of selecting at most one device to serve as bus master at any particular time. System control is concerned with making certain kinds of information available to all the devices in the system simultaneously, frequently for the purpose of forcing them into a common state. Stated thus, both functions seem relatively straightforward; but in fact both are intimately connected with the far more subtle issue of program control: the initiation and overall coordination of processes and algorithms which use the bus for transfer of data. (The term "intelligence", though sometimes used in this context, reflects far too narrow a view of human reasoning abilities.) Our discussion of arbitration and system control will therefore begin with a few comments about program control.

The design of UNIBUS is based on the underlying assumption, due orginally to von Neumann, that all program control will orginate from a single central processing unit (CPU). Under control of the CPU, UNIBUS is intended as a link primarily for the transfer of instructions and data between the CPU on the one hand, and memory and peripherals on the other. This assumption has several implications for UNIBUS.

The CPU is bus master by default, i.e., when no other device requests mastership. The arbitrator and the origin of the system control functions are physically located in the CPU; the CPU is the only device which can change its own priority for access to the bus. Most importantly, the ability of bus masters other than the CPU to affect the flow of program control is severely limited.

Brookhaven FASTBUS makes none of these assumptions, although there would be no problem in developing a von Neumann-like computer using FASTBUS. Interestingly, the two systems are superficially similar in assigning a distinct priority level to every potential master and in having a central arbitrator to grant bus mastership to the requesting device of highest priority. However, UNIBUS allows only two very specific modes of mastership to devices other than the CPU:

1. INTERRUPT, in which a device can alter the flow of program control by placing a single address on the bus which directs the CPU to the location of the interrupting routine in memory; and

2. DIRECT MEMORY ACCESS (DMA), in which a device may transfer data directly to or from memory or peripheral, without affecting program flow directly.

DMA transactions can occur at highest priority, precisely because they are "transparent" to program control. Interrupts, on the other hand, may occur at several levels of priority, and the processor may vary its own priority with respect to the various levels. Brookhaven FASTBUS, as noted above, neither defines any one device as the central processor nor imposes any restriction on the kinds of mastership allowed to non-processor devices.

In the area of system control, Brookhaven FASTBUS also maintains far greater flexibility, free of the von Neumann orthodoxy. Brookhaven FASTBUS also allows for far greater variety of system architecture than does UNIBUS. Despite the best efforts of certain DEC-compatible manufacturers, it is difficult to link more than one UNIBUS segment into a coherent system, because the original design gave little if any thought to multi-bus systems. Brookhaven FASTBUS, by contrast, is designed to support tree-like structures in which both the branches and nodes consist of individual FASTBUS segments.[2] The computer interface sits at the root node, and may issue system control commands for its own and other FASTBUS segments in the system. The BROADCAST bit, when enabled, permits a source of system control information to use the entire data space and forces all devices on a single bus segment ("local" broadcast) or on all bus segments ("global" broadcast) to respond.[3] Broadcast commands, however, need not originate from the interface, UNIBUS or its CPU; any device on FASTBUS may trigger a broadcast by writing into a special FASTBUS-addressable register in the interface.

By comparison, UNIBUS provides for only the most primitive system control functions: initialization of all devices into a well-defined starting state (also provided by Brookhaven FASTBUS) and indication of power- line status.

Based on our discussion of arbitration and system control, the following tasks emerge as functions the interface must perform:

1. Gain mastership on UNIBUS for interrupt and DMA transactions initiated from FASTBUS.

2. Gain mastership on FASTBUS to complete UNIBUS-initiated data transfers.

3. Issue local and global broadcast commands initiated by either bus.

IV. Putting It Together: the Design of the Interface

The basic design goals and requirements of the interface have already been outlined in the previous section. Once the problem was clearly understood, finding a solution was fairly straightforward. In this section we will outline briefly the overall architecture of the interface. We will then focus on the three most challenging aspects of the design: address mapping, status and error logging, and interrupt handling.

The guiding design principles are as follows:

1. For simplicity, there are no internal data registers, except as needed to guarantee the integrity 32-bit word transfers; wherever possible, data is transferred by clamping the two busses together.

2. Consistent with #1, minimum overhead use is made of either bus; for example, a 32-bit transfer to or from Brookhaven FASTBUS will occupy it about the same time as one of the two corresponding 16-bit transfers on UNIBUS.

3. Within the context of a simple system, maximum attempt is made to preserve status and error information, and make them available to either bus.

4. Data transfers initiated by FASTBUS normally gain control of UNIBUS via DMA. The interrupt pathway is reserved for special 16-bit FASTBUS messages and for various categores of error messages.

5. The interface is selected as the most convenient location for the central arbitrator of its own FASTBUS segment.

A simplified block diagram of the interface, showing all data and address pathways, is shown in Figure 4. Address transformations from UNIBUS to Brookhaven FASTBUS are performed by the address map, which will be described in detail below. In the opposite direction, they are handled by a shift matrix and counter which produces the correct address-to-data mapping and generates sequential addresses for block transfers. Every 32-bit transfer on Brookhaven FASTBUS will correspond to two 16-bit transfers on UNIBUS. In order to avoid tying up FASTBUS between UNIBUS words, temporary storage is provided in the high-word latch (HWL) and low-word latch (LWL), for data originating from FASTBUS and UNIBUS, respectively. Multiplexing of high- and low-order 16-bit words, composition, decomposition and byte shifting are performed by a combinational data multiplexer. Multiplexing of address and data, and bidirectional multiplexing are performed on an internal bus. A FASTBUS interface section does TTL/ECL level shifting and bidirectional buffering between the internal bus and the FASTBUS address/data lines. Two registers are included to handle broadcast commands: the FASTBUS broadcast register (FBR) for FASTBUS-initiated commands and the UNIBUS-addressable Low-order Broadcast Register (LBR). The high-order 16-bits of a UNIBUS-initiated broadcast are transmitted directly to FASTBUS by writing to a dummy location; the contents of the LBR are appended on to the explicit high-order part.

Not shown in Figure 4 are two control sections, which govern all data transfer control bits, requests for mastership, and control of multiplexers, drivers, address counters and data latches. One of these circuits controls FASTBUS-initiated transfers and is triggered by synchronization bits issued by the Brookhaven FASTBUS master; the other plays a corresponding role during UNIBUS-initiated transfers. Each is implemented using a Moore-type machine with pseudo-synchronous clock-input control.[4]

We turn next to the problem of address mapping from UNIBUS to Brookhaven FASTBUS. Only 4096 locations (12 bits) of the 18-bit UNIBUS address space are allotted to Brookhaven FASTBUS.

The transformation from 12 to 32 bits is accomplished by programming each of 16 32-bit mapping registers with a base address on Brookhaven FASTBUS. The 4 highest order bits of the 12 then select one of the mapping registers. The remaining 8 bits are shifted 0, 1 or 2 bits to the right to implement the proper address-to-data mapping and the result is logically OR'ed with the selected base address to form the actual Brookhaven FASTBUS address. This scheme allows the programmer to reserve 16 blocks of address space on Brookhaven FASTBUS, each with a capacity of 64 contiguous 32-bit words, 128 16-bit words or 256 8-bit bytes. In addition to the 32-bit base addresses, each mapping register also contains 5 bits of control information, including a bit for selecting 16 or 32-bit transfers.

The second topic which bears more detailed examination is the area of status and error logging. There are two registers, shown in Fig. 4, which are used for this purpose: the interrupt register (IR) and control/status register (CSR). Four CSR bits answer the following questions about the most recent transaction in the interface:

1. Was it initiated by FASTBUS or UNIBUS?

2. Was it part of a block transfer?

3. Was it a READ or WRITE operation?

4. Is the second UNIBUS word pending of a 32-bit transfer?

For a transaction which does not terminate in the usual way, additional information is stored in the IR. The four high-order bits contain a flag code indicating the type of error or anomaly which occurred; the low-order 12 bits store the UNIBUS address of the transaction in question. The flag codes distinguish among the following types of occurrence:

1. Failure of FASTBUS slave to return address handshake within 4 msec.

2. Failure of FASTBUS slave to return data handshake within 3 msec.

3, 4, 5. Presence of BUSY bit, EMPTY bit, or both when slave returns data handshake.

6. Disagreement of the two UNIBUS words of a 32-bit transfer as to READ/WRITE; i.e., alignment error.

7. Failure of any word transfer on either bus to terminate within 10 msec; includes arbitration failure.

8. Last word of DMA block transfer.

The interrupt register is also used by masters on Brookhaven FASTBUS to send messages to UNIBUS; in this case, only the leading bit is a flag bit, and the remaining 15 bits are available for data. This type of event will be referred to as Case #9. In all 9 cases, an ERROR bit is set in the CSR, and two additional CSR bits are used to distinguish between Cases #1-7, Case #8 and Case #9.

Any or all of these classes of event may trigger an interrupt, subject to the programming of 6 interrupt enable bits in the CSR. BUSY/EMPTY interrupts are enabled separately using a control bit in each of the 16 address mapping registers. A request for an interrupt causes the CSR INTERRUPT bit to be set, and the contents of the IR are locked in until the INTERRUPT bit has been cleared by the processor. Thus, if multiple interrupts occur, later ones may be locked out until information from the first has been read by the processor. However, multiple interrupts will cause an INTERRUPT OVERFLOW bit to be set in the CSR.

The appearance of an interrupt between the first and second UNIBUS words of a 32-bit transfer can corrupt the remainder of the transaction and cause loss of alignment for later transfers. An option was therefore included which allows the programmer to suspend interrupts until after the second word of a two-word UNIBUS transfer. This option is programmable separately in each of the 16 mapping register control fields.

V. Conclusion

Our aim has been to show that design of an interface between UNIBUS and Brookhaven FASTBUS was neither difficult to conceptualize nor hard to implement once the fundamentals of bus architecture were understood. The project lasted about seven months, occupying an electronic engineer full-time and an electronic technician half-time. Physically, the device is packaged in a 5-1/4" rackmount enclosure, not including power supplies. There are four quad-size UNIBUS cards, a modified UNIBUS backplane, two single-width standard DEC modules and a Brookhaven FASTBUS-size card. The interface is completely self-contained, and connects to both UNIBUS and Brookhaven FASTBUS via input and output cables, or cable and terminator. Total component cost was under $2000, and power requirements are approximately 10A @ +5v.; 10A @ -5v; and 2A @ -2v. The interface has been in operation for one year and is part of a CP violation experiment (Exp. 749) at the Brookhaven AGS.

Acknowledgments

This project would not have been possible without the encouragement of William P. Sims, on the technical support of Kathy Brown, Joan Depken, Paula Hughes, and Pat Oster; and their contributions are gratefully acknowledged.

References

1. J.V. Levy, "Buses, the Skeleton of Computer Structures" Computer Engineering, C.G. Bell, J.C. Mudge, J.E. McNamara, Eds., Digital Equipment Corp., Bedford, MA, Ch. 11 (1978).

2. L.B. Leipuner, et al. "A Five Segment Brookhaven FASTBUS System", in these Proc.

3. W.M. Morse, et al. "The Brookhaven Segment Interconnect", in these Proc.

4. F.J. Hill, G.R. Peterson, Introduction to Switching Theory and Logical Design, 3rd Ed., Sections 10.9 and 11.4 (John Wiley & Sons, NY 1981).

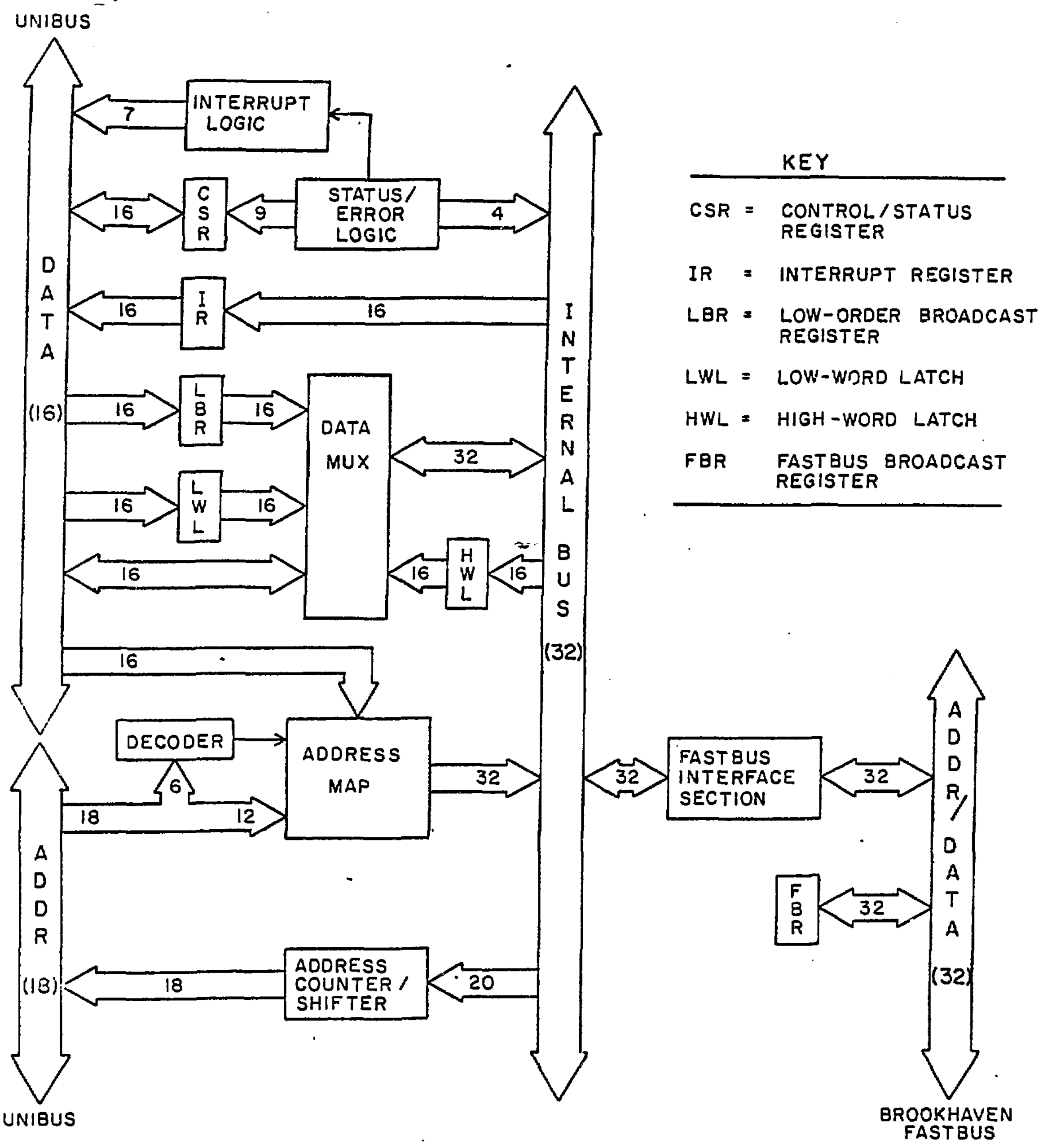

Fig. 4
Block Diagram